**Code Snippet**

**Data Set**

[**https://www.kaggle.com/datasets/focatak/malapi2019**](https://www.kaggle.com/datasets/focatak/malapi2019)

Given Frequency of the different malware.

|  |  |
| --- | --- |
| malware family | api\_calls |
| Spyware | 832 |
| Downloader | 1001 |
| Trojan | 1001 |
| Worms | 1001 |
| Adware | 379 |
| Dropper | 891 |
| Virus | 1001 |
| Backdoor | 1001 |

**Bar chart**

import matplotlib.pyplot as plt

malware\_family = ['Spyware', 'Downloader', 'Trojan', 'Worms', 'Adware', 'Dropper', 'Virus', 'Backdoor']

api\_calls = [832, 1001, 1001, 1001, 379, 891, 1001, 1001]

plt.bar(malware\_family, api\_calls)

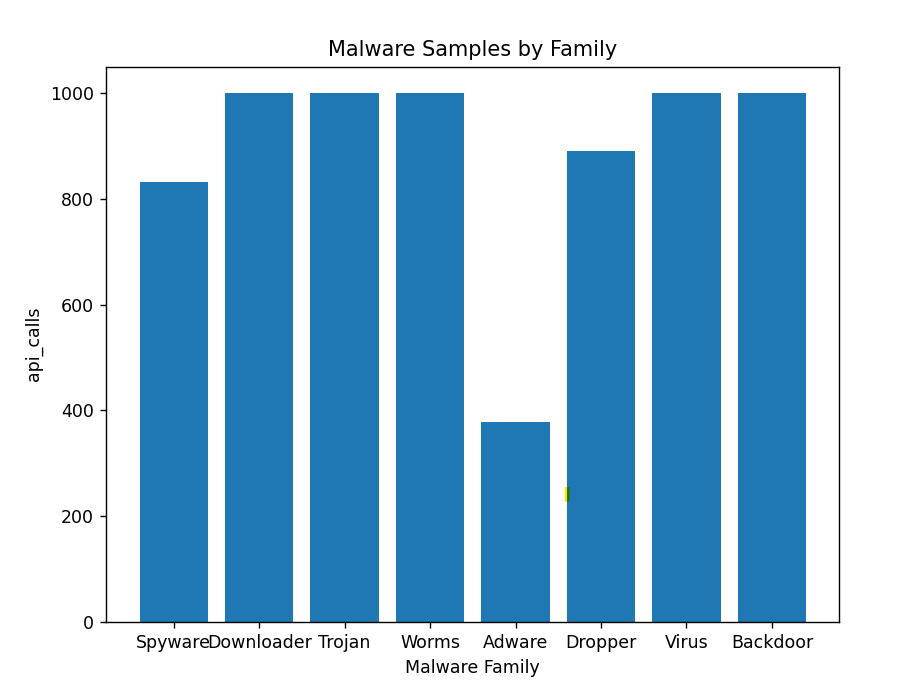
plt.title('Malware Samples by Family')

plt.xlabel('Malware Family')

plt.ylabel('api\_calls')

plt.show()

**Output**



**Range & Standard Deviation**

import pandas as pd

import numpy as np

# Define data

data = pd.DataFrame({

    'Malware Family': ['Spyware', 'Downloader', 'Trojan', 'Worms', 'Adware',

    'Dropper', 'Virus', 'Backdoor'],

    'api\_calls': [832, 1001, 1001, 1001, 379, 891, 1001, 1001]

})

# Calculate range

data\_range = np.max(data['api\_calls']) - np.min(data['api\_calls'])

print('Range:', data\_range)

# Calculate standard deviation

data\_std = np.std(data['api\_calls'])

print('Standard Deviation:', data\_std)

Output:

Range: 622

Standard Deviation: 201.88792528281627

**Box Plot**

import matplotlib.pyplot as plt

import pandas as pd

data = pd.DataFrame({

    'Malware Family': ['Spyware', 'Downloader', 'Trojan', 'Worms', 'Adware', 'Dropper', 'Virus', 'Backdoor'],

    'api\_calls': [832, 1001, 1001, 1001, 379, 891, 1001, 1001]

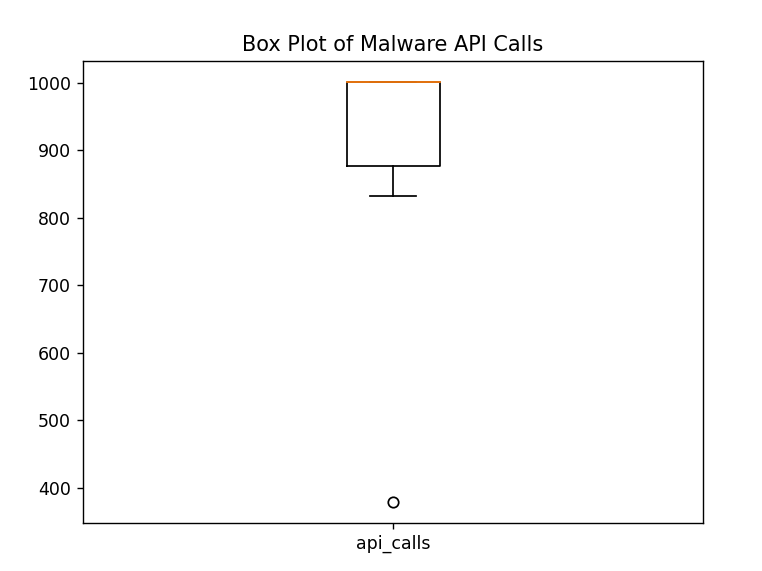
})

plt.boxplot(data['api\_calls'])

plt.xticks([1], ['api\_calls'])

plt.title('Box Plot of Malware API Calls')

plt.show()



**Generate One Hot Encoding using Python for the Given dataset label.csv**

import pandas as pd

# Load the data into a pandas DataFrame

df = pd.DataFrame({

    'MalwareFamily': ['Spyware', 'Downloader', 'Trojan', 'Worms', 'Adware', 'Dropper', 'Virus', 'Backdoor'],

    'api\_calls': [832, 1001, 1001, 1001, 379, 891, 1001, 1001]

})

# Generate one-hot encoding for the malwarefamily column

one\_hot\_encoded = pd.get\_dummies(df['malwarefamily'])

# Combine the one-hot encoded columns with the original DataFrame

df\_encoded = pd.concat([df, one\_hot\_encoded], axis=1)

# Print the encoded DataFrame

print(df\_encoded)

**Output:** It will generate the below data and then import the same into 'one\_hot\_encoding.csv' file.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Malware Family** | **api\_calls** | **Spyware** | **Downloader** | **Trojan** | **Worms** | **Adware** | **Dropper** | **Virus** | **Backdoor** |
| Spyware | 832 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Downloader | 1001 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Trojan | 1001 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Worms | 1001 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Adware | 379 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Dropper | 891 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Virus | 1001 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Backdoor | 1001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Linear Regression** (For output refer to the section Linear Regression under Finding & Analysis)

* **Spyware**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Spyware'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Spyware')

plt.title('Linear Regression: API Calls vs. Spyware')

plt.show()

* **Downloader**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Downloader'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Downloader')

plt.title('Linear Regression: API Calls vs. Downloader')

plt.show()

* **Trojan**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Trojan'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Trojan')

plt.title('Linear Regression: API Calls vs. Trojan')

plt.show()

* **Worms**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Worms'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Worms')

plt.title('Linear Regression: API Calls vs. Worms')

plt.show()

* **Adware**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Adware'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Adware')

plt.title('Linear Regression: API Calls vs. Adware')

plt.show()

* **Dropper**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Dropper'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Dropper')

plt.title('Linear Regression: API Calls vs. Dropper')

plt.show()

* **Virus**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Virus'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Virus')

plt.title('Linear Regression: API Calls vs. Virus')

plt.show()

* **Backdoor**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Backdoor'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Backdoor')

plt.title('Linear Regression: API Calls vs. Backdoor')

plt.show()

**Calculating Mean Squared Error, Root Mean Squared Error (RMSE) and R-squared (R2) score:**

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

# Load the data

data = pd.read\_csv('one\_hot\_encoding.csv')

# Split the data into features and target variable

X = data.iloc[:, 2:]

y = data.iloc[:, 1]

# Create a linear regression model and fit the data

model = LinearRegression()

model.fit(X, y)

# Predict the target variable using the trained model

y\_pred = model.predict(X)

# Calculate the mean squared error (MSE)

mse = mean\_squared\_error(y, y\_pred)

# Calculate the root mean squared error (RMSE)

rmse = np.sqrt(mse)

# Calculate the R-squared (R2) score

r2 = r2\_score(y, y\_pred)

print("Mean Squared Error (MSE): ", mse)

print("Root Mean Squared Error (RMSE): ", rmse)

print("R-squared (R2) score: ", r2)

plt.bar(["r2 score"], [r2])

plt.ylim(0, 1)

plt.title("r2 score for linear regression model")

plt.show()

**Output:**

**Mean Squared Error (MSE):** 9.244463733058732e-33

**Root Mean Squared Error (RMSE):** 9.614813431917819e-17

**R-squared (R2) score:** 1.0

